University of Maryland College Park
Dept of Computer Science
CMSC132, Midterm #2 Key
Summer 2008

First Name (PRINT): _______________________________________________________

Last Name (PRINT): _______________________________________________________

University ID: ___________________________________________________________

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: __________________________________________________________

Instructions

➢ This exam is a closed-book and closed-notes exam.
➢ Total point value is 100 points, 50 minutes exam.
➢ Please use a pencil to complete the exam.
➢ PUNT RULE: For any question, you may write PUNT, and you will get ¼ of the points for the question (rounded down). If you feel totally lost on a question, you are encouraged to punt rather than write down an incorrect answer in hopes of getting some partial credit.
➢ WRITE NEATLY. If we cannot understand your answer, we will not grade it (i.e., 0 credit).

Grader Use Only

<table>
<thead>
<tr>
<th>#1</th>
<th>Software Engineering</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>#2</td>
<td>OO Design (UML)</td>
<td>20</td>
</tr>
<tr>
<td>#3</td>
<td>GUI/Inner Classes</td>
<td>12</td>
</tr>
<tr>
<td>#4</td>
<td>Heaps</td>
<td>14</td>
</tr>
<tr>
<td>#5</td>
<td>Binary Trees</td>
<td>24</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>
Problem 1 (30 pts) Software Engineering

1. T The Waterfall model begins a new step only when the previous step is complete.
2. T Maintenance is considered a component of the software life cycle.
3. T Software process models are codified sets of practices for software development.
4. T The Waterfall model is more appropriate for small software projects.
5. F Pair programming is a practice associated with the Waterfall model.
6. F Abstraction and encapsulation help make programs run faster.
7. T In OO systems invoking an object’s method is equivalent to sending the object a message.
8. F Coding is the largest component of software development.
9. F We prefer inheritance over composition when designing a system.
10. F Encapsulation makes code modification harder and reduces code reuse.
11. T Formal methods are mathematically-based techniques for specification, development, and verification of software systems.
12. T Some people believe recent software development techniques have reduced the cost of change.
13. F Regression testing ensures functionality is not lost when software is bought & sold.
14. T In the Blackboard architecture components communicate through a shared updatable blackboard.
15. T A mock object is similar to a stub but it can record calls made to the object.
16. T Test coverage measures whether code is executed by some test case.
17. The waterfall model of software development emphasizes predictability.
18. The iterative model of software development emphasizes adaptability.
19. In OO design, objects in a system correspond to nouns in the problem description.
20. Similarly, interactions between objects correspond to verbs in the problem description.
21. (4 pts) The software life cycle is a sequence of essential operations necessary for producing quality software. Mention four of those essential operations.

   Answer: Any four of: Problem Specification, Program Design, Algorithms and Data Structures, Coding and Debugging, Testing and Verification, Documentation and Support, Maintenance
22. (2 pts) What is clear box testing? Briefly explain.

   Answer: Testing where we are allowed to examine the code.
23. (2 pts) What is black box testing? Briefly explain.

   Answer: Testing where we are not allowed to examine the code. Tests behavior in response to inputs.
24. (2 pts) What is regression testing? Briefly explain.

   Answer: Testing to see if some functionality has been lost.
Problem 2 (20 pts) OO Design (UML)

Draw a UML class diagram for two types of phones a company sells: regular phones and camera phones. A regular phone has an assigned number, a charger, and allows you to dial any number in the USA. A camera phone is a regular phone with the capability of taking pictures. Every charger has a manufacturer id. You don’t need to represent get/set methods or constructors in your diagram.

One Possible Design:

---

Problem 3 (12 pts) GUI/Inner Class

1. F We can only register one listener object for a button.
2. T The Event Dispatching Thread is in charge of updating GUI components.
3. T JFrame and JApplet are examples of containers.
4. F In the Blackjack project you were asked to implement the view component of the MVC paradigm.
5. F A program that reads two values, computes the average and ends is an example of an event-driven program.
6. (7 pts) Answer the following questions based on the classes that follow.
   a. F The Tetris class is considered an inner class.
   b. F An inner class may not access private methods of the outer class (containing class).
   c. F We can only have one reference variable associated with an anonymous inner class object.
   d. Complete the assignment below by assigning to megaTetris an object that extends the Tetris class and overrides the playGame() method so that the method will print “Playing MegaTetris”. You must use an anonymous inner class to define the object.
package innerClass;

class Tetris {
    public void playGame() { System.out.println("Playing Tetris"); }
}

public class Driver {
    public static void main(String[] args) {
        Tetris megaTetris = /* COMPLETE THIS ASSIGNMENT */
    }
}

Answer:

    Tetris megaTetris = new Tetris() {
        public void playGame() { System.out.println("Playing MegaTetris"); }
    };

Problem 4 (14 pts) Heaps

Use the following heap to answer the questions that follow.

1. (2 pts) Draw the heap as an array.

   Answer:

   7 10 8 13 20
2. (3 pts) Draw the heap that would result from inserting 3 in the above heap.

Answer:

```
  3
 /  \\   \\   \\
 10  7  8
 /  \\  \\  \\
13 20
```

3. (3 pts) Draw the heap that would result by deleting 7 from the original heap.

Answer:

```
  8
 /  \\   \\   \\
 10 20
 / \\
13
```

4. (6 pts) For a heap (designed to find the MAXIMUM value of a collection):

   a. F Every heap is a binary search tree.
   b. F A post-order traversal of a heap will list its values in increasing order.
   c. T The heap will always be a balanced-tree (elements can be inserted in any order)
   d. T The time required to find the maximum element is O(1).
   e. T A sorted array represents a heap.
   f. T Priority queues can be implemented using heaps.
Problem 5 (24 pts) Binary Trees

Implement the method below based on the following Java class definitions. You may not add any instance variables or static variables to either class and you may not add any methods to the Node class. **Non-recursive solutions will receive zero credit.**

```java
public class BinarySearchTree <K extends Comparable<K>, V> {
    private class Node {
        private K key;
        private V data;
        private Node left, right;
        public Node(K key, V data) {
            this.key = key;
            this.data = data;
        }
    }

    private Node root;
    public V find(K key) { /* YOU MUST IMPLEMENT THIS METHOD */ }
    public void removeLeaves() { /* YOU MUST IMPLEMENT THIS METHOD */ }
}
```

1. **(12 pts) Implement the method find** that returns the element associated with the key parameter or null otherwise.

**Answer:**

```java
public V find(K key) {
    return findAux(key, root);
}

public V findAux(K key, Node rootAux) {
    if (rootAux == null)
        return null;
    else {
        int comparison = key.compareTo(rootAux.key);
        if (comparison==0)
            return rootAux.data;
        else if (comparison < 0)
            return findAux(key, rootAux.left);
        else
            return findAux(key, rootAux.right);
    }
}
```
2. (12 pts) Implement the method `removeLeaves` that removes all the leaf nodes of the tree.

**Answer:**

```java
public void removeLeaves() {
    if (root == null)
        return;
    else if (root.left == null && root.right == null)
        root = null;
    else {
        removeLeavesAux(root.left, root);
        removeLeavesAux(root.right, root);
    }
}

private void removeLeavesAux(Node rootAux, Node parent) {
    if (rootAux.left == null && rootAux.right == null)
        if (rootAux == parent.left)
            parent.left = null;
        else
            parent.right = null;
    else if (rootAux.left == null && rootAux.right != null)
        removeLeavesAux(rootAux.right, rootAux);
    else if (rootAux.left != null && rootAux.right == null)
        removeLeavesAux(rootAux.left, rootAux);
    else {
        removeLeavesAux(rootAux.left, rootAux);
        removeLeavesAux(rootAux.right, rootAux);
    }
}
```